UNITED STATES PATENT APPLICATION

for

SYSTEMS AND METHODS FOR FAST RANDOM ACCESS AND BACKWARD PLAYBACK OF VIDEO FRAMES USING DECODED FRAME CACHE

Inventor:

Boon-Lock Yeo

Docket No.: 42390.P7327

Prepared by:
Alan K. Aldous
Reg. No. 31,905

"Express mail" label no. EL034432901US

SYSTEMS AND METHODS FOR FAST RANDOM ACCESS AND BACKWARD PLAYBACK OF VIDEO FRAMES USING DECODED FRAME CACHE

Background of the Invention

Technical Field of the Invention: The present invention relates to systems and methods for random access and backward playback of video frames.

Background Art: Video recording has become extremely popular. Traditionally, video tape has been used to record video streams. A video stream is a sequence of frames. In many cases, a VCR (video cassette recorder) is used to playback the video tape. More recently, computers, such as personal computers, have been used to play video streams.

There are various formats of digital video signals. However, popular digital video formats include MPEG (Moving Picture Experts Group) formats. Current and proposed MPEG formats include MPEG-1 ("Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 MBits/s," ISO/IEC JTC 1 CD IS-11172 (1992)), MPEG-2 ("Generic Coding of Moving Pictures and Associated Audio," ISO/IEC JTC 1 CD 13818 (1994); and MPEG-4 ("Very Low Bitrate Audio-Visual Coding" Status: ISO/ IEC JTC 1/SC 29/WG 11, 3/1999). There are different versions of MPEG-1 and MPEG-2.

Although video streams are typically played forward, techniques have been developed to play them in the backward direction. Frame-by-frame backward playback refers to playing each frame in a backward sequence (rather than skipping some). A difficulty in playing in the backward direction is that decoding a frame may require use of another decoded frame which has not yet been decoded. One solution is to decode only certain independent frames or a limited number of other frames. This technique is not frame-by-frame and is unsatisfactory in many cases because it results in a loss of detail. Another technique is to decode each frame in a group of pictures before they are needed and then play them in use them as needed. However, this is wasteful in that not all of them might be needed. Still another technique is to decode the frames as needed, but to redecode the frames rather than store them in decoded form for further use. This is likewise wasteful because frames may be repeatedly decoded over a short amount of time.

Accordingly, there is a need for an effective technique for frame-by-frame backward playback.

Summary of the Invention

In some embodiments, the invention includes a method of processing a video stream. The method involves detecting a request to playback a particular frame. It is determined whether a decoded version of the particular frame is in a decoded frame cache. If it is not, the method includes(i) determining a frame dependency for the particular frame; (ii) determining which of the frames in the frame dependency are in the decoded frame cache; (iii) decoding any frame in the frame dependency that is not in the decoded frame cache and placing it in the decoded frame cache; and (iv) using at least some of the decoded frames in the frame dependency to decode the particular frame to create a decoded version of the particular frame.

In some embodiments, the request to playback a particular frame is part of a request to perform frame-by-frame backward playback and the method is performed for successively earlier frames with respect to the particular frame as part of the frame-by-frame backward playback.

In some embodiments, the part (i) is performed whether or not it is determined that a decoded version of a particular frame is in the decoded frame cache without part (iv) being performed.

Other embodiments are described and claimed.

Brief Description of the Drawings

The invention will be understood more fully from the detailed description given below and from the accompanying drawings of embodiments of the invention which, however, should not be taken to limit the invention to the specific embodiments described, but are for explanation and understanding only.

FIG. 1 is a block diagram representation of a computer system that may be used in connection with some embodiments of the invention.

FIG. 2 is a high level schematic block diagram representation of come components of the computer system of FIG. 1.

FIG. 3 is an illustration of frames in the order of forward display.

Detailed Description

Overview

The invention relates to systems and methods for random access and backward playback of video frames.

Referring to FIG. 1, a computer system 10 is illustrated as a desk top personal computer, but may be any of a variety of other computers including a portable computer (e.g., laptop), a set top box and a television, a network computer, a mainframe computer, etc. Computer 10 includes chassis 14, a monitor display 16, a mouse 18, and a keyboard 22. Display 16 may be integrated to chassis 14. A video source 24 provides video streams to electronic components in chassis 14. The video streams may be provided in a variety of ways including through a serial conductor directly from a camera, a disc, a modem directly from another computer or through the Internet, or other means.

FIG. 2 illustrates some of the components in chassis 14 in a high level schematic form. Referring to FIG. 2, a memory 30 represents a variety of memory including, for example, a hard drive, main memory, video memory (e.g., video static random access memory (VSRAM)), and a compact disc (CD), if used (which are examples articles including a computer readable medium). For example, memory 30 includes software 32 and video stream signals 34. Memory 30 may also include a database 36 to hold video document detail information and associated query and display software. Software, video stream signals, and other signals, such as control signals, are processed by a processor 38 with the assistance of video processing circuitry 40. Input/output (I/O) circuitry 42 interfaces between the other components of FIG. 2 and, for example, user input devices (e.g., mouse 18 and keyboard 22) and display 16. Examples of user interface devices include a keyboard and one or more other user input devices including, for example, a mouse, joystick, trackball, keyboard, light pen, touch pad, touch screen, gesture recognition mechanism, etc. Again, FIG. 2 is intended to be a high level schematic representation and the invention is not restricted to the particular details therein.

The concepts of random access and frame-by-frame backward playback can be understood with reference to FIG. 3, which represents the order of forward playback of frames F1, F2, ... F26, ..., which are each part of the same video stream. In forward playback, the order the frames are displayed on display 16 is that shown in FIG. 3, namely, F1, then F2, then F3, etc. In frame-by-frame backward playback, successive lower numbered frames are displayed. Backward playback may be initiated through a user input device or other means (e.g., software control). For example, assume that when backward playback is initiated, the current frame to be displayed is frame F10. Then, for frame-by-frame backward playback, the next frames to be displayed are F9, F8, F7, etc. until something indicates that displaying frames of the video stream should stop or the direction is changed. There is backward playback that is other than frame-by-frame backward playback in which not every frame is displayed (e.g., only I frames or only I frames and P frames). For random access playback, the frame displayed can be any frame, rather than merely then next frame. For example, then order of display could be F10, F11, F12, F15, F16, F6, F7. As used herein, random does not mean unpredictable, but rather that any frame within the video stream or a section of the video stream can be accessed. The first frame of a frame-by-frame backward playback is an example of random access.

The invention is not restricted to any particular digital format, but will be described in terms of MPEG video formats. Various formats other than MPEG may be used. The video signals may have interleaved or non-interleaved frames. The invention is applicable to formats having dependencies and those that to not.

<u>Fast Random Access of Frames and Backward Frame-by-frame Playback</u>

MPEG-1 and MPEG-2 video are made up of three basic frame types: I frames, P frames and B frames. I frames are coded independently of other frames. P frames are coded based on the previous I or P frames. B frames are also known as Bi-directional frames and are coded based on the previous and/or next I or P frames. For MPEG-2 video coded using field coding, B frames may also depend on a different field coded as a B frame. The costs of decompression varies across the different frame types. I frames are cheapest to decode, followed by P frames and then by B frames. To decode P and B frames, motion compensation is required. A B frame is

typically more expensive to decode than a P frame because it may depend on two frames whereas a P frame only depends on one frame.

For convenience, the following examples will be described with a 9 frame group of pictures (GOP) (which has 10 frame if the I frame of the next GOP is included). However, the invention is not limited to a particular number of frames in the GOP. For example, a GOP could typically have 15 or 30 frames, or other some other number of frames.

The invention is not restricted to use with frames with any particular resolution or number of bytes. For example, for MPEG1 video (352x240 resolution), the size of one decompressed frame may be 1/4 Megabyte (Mbyte) in RGB mode and 1/8 MByte in YUV mode. With greater resolutions, the size can be much greater.

Consider the following pattern of the GOP is the order of displaying frames: I1 (F1), B1 (F2), B2 (F3), P1 (F4), B3 (F5), B4 (F6), P2 (F7), B5 (F8), B6 (F9), I2 (F10). The frame numbers are in parenthesis and the numbers after the frame types are used to distinguish among the different frames with the same encoding type. In contrast to the display order, the encode and decode orders are I1 P1 B1 B2 P2 B3 B4 I2 B1 B2.

1. <u>Random Access</u>

Random access into arbitrary frames of an MPEG video is not straightforward because of frame dependencies. For instance, to access P1, I1 needs to be first decoded; to access B4, which depends on P1 and P2, we will need to first decode I1, P1 and P2.

One approach would be to decode every frame in the GOP so that the needed decoded frames would be available. However, that brute force approach is wasteful. A better approach for random access is to maintain a list of immediate frame dependencies for each frame. The immediate frame dependencies specify the set of frames directly needed for decode operation of the current frame. For the above example, the following are immediate frame dependencies:

I1: none
B1: I1, P1
B2: I1, P1
P1: I1
B3: P1, P2
B4: P1, P2
P2: P1

B5: P2, I2
B6: P2, I2
I2: none

(The frame dependencies could be provided by a look-up table, which could be accessed by indices or frame numbers.)

Thus, decoding B5, involves using the decoded P2 and I2 are needed for motion compensation. In addition, decoding P2 involves using the decoded P1, which in return requires the decoded I1. Decoding B5 involves using decoded I1, I2, P1 and P2. Decoding B2 involves using the decoded I1 and P1 for motion compensation; decoding P1 in turns requires decoded I1. B2 therefore requires the decoded I1 and P1. Accordingly, the needed decoded frames are decoded first and stored in memory. Note that in some cases decoded frames will be stored even if they are not going to be displayed themselves, so they will be available in decoding other frames.

2. Frame-by-frame Backward Playback

Backward (reverse) playback of MPEG video can be straightforwardly implemented using random access techniques. Thus, to access the above 10 frames backward, we could use the random access method above to decode frame 10, then use the random access method to decode frame 9 without taking advantage of the fact that it was already used for decoding frame 10, and so on. This approach, however, does not exploit the temporal coherence of backward decoding. The following is a novel technique to exploit the coherence.

The decoded frames are stored in a decoded frame cache. Various types of memory may be used for the decoded frame cache. Main memory dynamic random access memory (DRAM) is an example. Video random access memory (VRAM) may also be used. A separate memory or section of memory could be dedicated solely to holding the decoded frames. The decoded frame cache does not have to be all in contiguous locations.

The decoded frames cache may be a fixed or variable size. If it is of fixed size it should be large enough to hold the minimum number of decoded frames needed considering the GOPs that may be encountered. The size could dynamically change if the number of frames in the GOP changes. Under one approach, the decoded frames cache is of a fixed size and a when the

cache is full, a Least Recently Used (LRU) replacement policy is used to replace the frame that has been least recently accessed. If the cache is not of a fixed size, it could hold a fixed number of frames and a LRU replacement policy could be used.

Using the previous example for a backward decode from frame 10 to 1, the following will happen using the new algorithm on frames 10 to 7:

Frame 10 is the I2 frame. I2 is decoded and stored in the decoded frames cache. Cache = [I2].

Frame 9 is the B6 frame. B6 needs I2, P2, P1 I1. P2, P1, and I1 are decoded. B6 is also decoded. I2 is already in the cache so it does not need to be re-decoded. Decoded P2, P1, I1 and B6 are stored in the cache. Cache = [I2, I1, P1, P2, B6].

Frame 8 is the B5 frame. B5 needs I2 and P2, which are already in the cache. Decode B5 and put in the cache. Cache = [I2, I1, P1, P2, B6, B5]

Frame 7 is the P2 frame. P2 needs P1 which is already decoded. Decode P2 and put in cache. Cache = [I2, I1, P1, P2, B6, B5].

Random access can also be more effectively performed using the above described frame caching technique used in backward playback. The key is to use the same caching mechanism for storing recently decoded frames and to re-use these frames if they are requested in the near future. For instance, the following set of frames may be requested to be decoded: I1, B3, B5. To decode B3, both P1 and P2 are needed. As a result, P1, P2 and I1 will be decoded and placed in the decoded frame cache and used from the decoded frame cache if they were already there. In the next request to decode B5, which depends on P2 and I2, only I2 needs to be decoded since P2 is already in the cache.

The caching technique can be performed through hardware or software control. The technique is described in terms of software pseudo code, but can be implemented in hardware or through software according to different pseudo code. That is, there are a variety of ways to implement the cache technique.

Consider the example I1 (F1), B1 (F2), B2 (F3), P1(F4), B3(F5), B4(F6), P2(F7), B5(F8), B6(F9), I2 (F10), mentioned above.

Assume there are the following two functions (1) DecodeCurrentFrame (N, ReferenceSet) and (2) GetDependencyFrameIndex (N).

In DecodeCurrentFrame (N, ReferenceSet), frame N is decoded according to an MPEG algorithm using frame N and the ReferenceSet. The ReferenceSet is the set of referenced frames needed to decode frame N. For example, ReferenceSet for P1 is {frame 1}, ReferenceSet for B4 is {frame 4, frame 7}. A decoded frame N is returned by the function. The decoded frame may be in RGB, YUV, or another format. RGB is used in the example.

In GetDependencyFrameIndex (N), a list of the reference frames that are needed to decode current frame N is obtained. A list of frame index or indices is returned. For example, e.g., GetDependencyFrameIndex (5) = {7, 10}.

In the following pseudo code there is a distinction between the index and the actual frame. For example, 10 is an index and frame 10 is the actual frame. There is an array of a data structure called MPEGFrameCache, which is the decoded frame cache. MPEGFrameCache has two attributes: LastTimeUsed (for using in the LRU technique) and FrameRGB.

The following is pseudo-code (lines 1 - 22) to implement GetFrame() using the caching technique according to some embodiments:

```
1     frame GetFrame(N)
2             SetofDependencyIndex = GetDependencyFrameIndex (N)
3             SetofDependencyFrame = {}
4             /* decode frame in the dependency list if needed */
5             /* decoding also forces the frame to go to the decoded frames cache */
6             for each FrameIndex in SetofDependencyIndex do
7                     if frame FrameIndex NOT in MPEGFrameCache then
8                             /* this call is recursive */
9                             Insert GetFrame (Frame) to SetofDependencyFrame
10                    else
11                            Retrieve frame indicated by FrameIndex from MPEGFrameCache
12                            Insert frame indicated by FrameIndex to SetofDependencyFrame
13                            Update LastTimeUsed of frame indicated by FrameIndex in
                                      MPEGFrameCache
14                    end if
15            end for

16            currentFrame = DecodeCurrentFrame (N, SetofDependencyFrame)
```

```
        if MPEGFrameCache is full then
                Remove element from MPEGFrameCache with oldest LastTimeUsed
        endif
        Insert currentFrame to MPEGFrameCache
        return currentFrame
end
```

With respect to the example given above, the following sequence of events happens to decode backward from frame 10. Assume that MPEGFrameCache is initially empty.

```
+ GetFrame (10)
        GetDependencyFrameIndex (10) = {}
        DecodeCurrentFrame (10, {})
        MPEGFrameCache = {frame 10}

+ GetFrame (9)
        GetDependencyFrameIndex (9) = {7, 10}
        Since frame 7 is not in MPEGFrameCache, call GetFrame (7)
        + GetFrame (7)
                GetDependencyFrameIndex(7) = {4}
                Since frame 4 is not in MPEGFrameCache, call GetFrame (4)
                + GetFrame (4)
                        GetDependencyFrameIndex (4) = {1}
                        Since frame 1 is not in MPEGFrameCache, call GetFrame (1)
                        + GetFrame (1)
                                GetDependencyFrameIndex (1) = {}
                                DecodeCurrentFrame (1, {})
                                MPEGFrameCache = {frame 1, frame 10}
                        DecodeCurrentFrame (4, {1})
                        MPEGFrameCache = {frame 1, frame 10, frame 4}
                DecodeCurrentFrame (7, {4})
                MPEGFrameCache = {frame 1, frame 10, frame 4, frame 7}

        Frame 10 is already in MPEGFrameCache
        DecodeCurrentFrame (9, {7, 10})
        MPEGFrameCache = {frame 1, frame 10, frame 4, frame 7, frame 9}

+ GetFrame (8)
        GetDependencyFrameIndex (8) = {7, 10}
        Both Frame 7 and 10 are in the MPEGFrameCache
        DecodeCurrentFrame (8, {7, 10})
```

MPEGFrameCache = {frame 1, frame 10, frame 4, frame 7, frame 9, frame 8}

....

In the above trace, the LastTimeUsed attribute of MPEGFrameCache is not indicated. However, the LRU technique could be used. Note that the invention is not limited to an LRU technique.

The frames in MPEGFrameCache are not necessarily ordered.

Rather than use a recursive call, the above listed pseudo code (lines 1 - 20) could be modified to include a loop wherein the terminating condition is that all frames on which frame N is dependent have been decoded and are available for use in decoding frame N.

Additional Information And Embodiments

Reference in the specification to "some embodiments" or "other embodiments" means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least some embodiments, but not necessarily all embodiments, of the invention. The various appearances of the term "some embodiments" in the description are not necessarily all referring to the same embodiments.

The term "responsive" and related terms mean that one signal or event is influenced to some extent by another signal or event, but not necessarily completely or directly. If the specification states a component, event, or characteristic "may", "might" or "could" be included, that particular component, event, or characteristic is not required to be included.

Those skilled in the art having the benefit of this disclosure will appreciate that many other variations from the foregoing description and drawings may be made within the scope of the present invention. Accordingly, it is the following claims including any amendments thereto that define the scope of the invention.